Chapter 01
程式設計起點：輸入和輸出

小到一個程式，大到一個軟體系統，都是「輸入 - 處理 - 輸出」的模式。所以，本章展開輸入和輸出的練習，讓讀者瞭解和實作如何編寫、編譯和偵錯程式，以及線上提交程式的基本過程。

對於 C 語言，scanf 函式和 printf 函式分別是輸入函式和輸出函式，宣告在標頭檔 stdio.h 裡。所以，在使用 scanf 函式和 printf 函式時，要加上「#include <stdio.h>」。

C++ 的輸出和輸入是用「串流」（stream）的方式實作，串流物件 cin、cout 和串流運算元的定義等資訊，在 C++ 的輸入 / 輸出串流程式庫中。因此，如果在程式中使用 cin、cout 和串流運算元，就要加上「#include<iostream>」。

1.1 輸出

程式設計學習的起點是，編寫一個在標準輸出中直接輸出一行字串「Hello World」的程式。「1.1.1 Fibonacci Sequence」是一道類似的試題。

1.1.1 ► Fibonacci Sequence

費氏數列是一個自然數的序列，定義如下：

- $F_1=1$；

碁峰 www.gotop.com.tw

- $F_2=1$；
- $F_n=F_{n-1}+F_{n-2}$，其中 $n>2$。

編寫程式，輸出費氏數列中的前 5 個數字。

輸出

輸出 5 個整數，即費氏數列中的前 5 個數字。在輸出中，任何兩個相鄰數字都用一個空格分隔，行的末尾沒有額外的空格或符號。

範例輸入	範例輸出
（無輸入）	1 1 2 3 5

試題來源：2019 ICPC Asia Yinchuan Regional Programming Contest

線上測試：計蒜客 A2268

❖ 試題解析

本題要求完成一個最簡單的程式，沒有輸入，只輸出費氏數列中的前 5 個數字。

參考程式 1 為 C 語言版，參考程式 2 為 C++ 語言版。

❖ 參考程式 1

```
01  #include <stdio.h>
02  int main(){
03      printf("1 1 2 3 5");          // 輸出費氏數列中的前 5 個數字
04  }
```

❖ 參考程式 2

```
01  #include<iostream>
02  using namespace std;
03  int main(){
04      cout<<"1 1 2 3 5"<<endl;      // 輸出費氏數列中的前 5 個數字
05  }
```


碁峰 www.gotop.com.tw

1.2 輸入與輸出

在「1.1.1 Fibonacci Sequence」的基礎上，完成「1.2.1 A+B Problem」，體驗程式的「輸入 - 處理 - 輸出」模式。

1.2.1 ▶ A+B Problem

計算 $a+b$。

輸入

兩個整數 a 和 b（$0 \le a, b \le 10$）。

輸出

輸出 $a+b$ 的結果。

範例輸入	範例輸出
1 2	3

線上測試：POJ 1000，ZOJ 1000

❖ 試題解析

本題是一道練習「輸入 - 處理 - 輸出」模式的入門試題。

首先，根據試題描述中提供的資料的範圍，定義三個 int 型別的變數 a、b、c；然後，輸入兩個整數，指定給 a 和 b；接下來，透過指定述句，計算運算式 $a+b$，指定給變數 c；最後，輸出結果 c。

❖ 參考程式

```
01  #include <stdio.h>
02  int main(void) {
03      int a, b,c;
04      scanf("%d%d", &a, &b);     // 輸入兩個整數 a 和 b
05      c=a+b;                     // 處理：計算 a+b
```


碁峰 www.gotop.com.tw

```
06      printf("%d\n",c);          // 輸出 a+b
07      return 0;
08  }
```

「1.2.1 A+B Problem」的參考程式是 C 語言版，建議讀者在此基礎上，完成「1.2.1 A+B Problem」的 C++ 語言版的程式。

碁峰 www.gotop.com.tw

Chapter 02
程式設計基礎 I

1984 年，圖靈獎得主尼古拉斯·沃斯（Nicklaus Wirth）提出了著名公式「演算法 + 資料結構 = 程式」，其中，演算法是程式設計解決問題的方法；資料結構是現實世界中，要被處理的資訊在程式中的表示形式。從程式語言的角度來看，資料結構是由基本的資料型別（即整數、實數、字元）以及陣列、指標、結構組成的。而演算法則是透過循序結構、選擇結構、迴圈結構和函式來完成的。選擇結構包括 if 選擇結構和 switch 選擇結構，迴圈結構包括 while 迴圈結構、do while 迴圈結構和 for 迴圈結構。

由於各類程式語言的書籍已經汗牛充棟，有關程式語言，我們不再贅述，而是側重於如何編寫程式解決問題。

在第 1 章「輸入 - 處理 - 輸出」的實作基礎上，本章程式編寫訓練的重點是如何正確地處理輸入和輸出，以及掌握基本資料型別、循序結構、選擇結構、迴圈結構、陣列、字串，並運用它們來分析問題和解決問題。透過簡單運算的程式編寫練習，學生可以掌握 C/C++ 或 Java 等程式語言的基本語法，熟悉線上測試系統和程式設計環境，初步學會如何將一個自然語言描述的實際問題抽象成一個計算問題，得到運算過程。繼而編寫程式完成運算過程，並將運算結果還原成對原來問題的解答。

碁峰 www.gotop.com.tw

2.1 選擇結構

程式語言中的選擇結構包括 if 選擇結構和 switch 選擇結構。if 選擇結構有三種形式。

1. 單分支 if 選擇結構，例如：

```
if (score>=60) printf("pass");              // 單分支 if 述句
```

2. if else 選擇結構，例如：

```
if (score> =60) printf("pass");
     else printf("fail");
```

3. 多分支 if else 選擇結構，例如：

```
if (score>=90) printf("excellent");        // 多分支 if 述句
    else if (score>=80) printf("good");
        else if (score>=70) printf("secondary");
            else if (score>=60) printf("pass");
                else printf("fail");
```

首先提供的「2.1.1 Accurate Movement」，是一個單分支 if 述句實作。

2.1.1 ► Accurate Movement

Amelia 做了一個 $2\times n$ 大小的矩形盒子，裡面有兩條平行的軌道，每條軌道上都有一個矩形。短矩形的尺寸為 $1\times a$，長矩形的尺寸為 $1\times b$。長矩形的兩端各有一個止動欄杆，短矩形則始終位於這兩個止動欄杆之間。

只要短矩形在長矩形的止動欄杆之間，矩形就可以沿著軌道移動，一次可以移動一個矩形。因此，在每次移動時，Amelia 都會選擇其中一個矩形移動它，而另一個矩形則保持原來的位置。最初，兩個矩形在矩形盒子的一側對齊，Amelia 希望透過盡可能少的移動次數，將兩個矩形移動到矩形盒子的另一側並對齊，如圖 2.1-1 所示。要達到這一目標，Amelia 最少要移動矩形多少次？

碁峰 www.gotop.com.tw

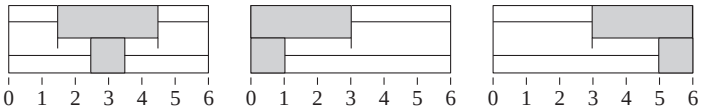

圖 2.1-1

輸入

輸入一行，提供三個整數 a、b 和 n（$1 \le a < b \le n \le 10^7$）。

輸出

輸出一行，提供一個整數，即 Amelia 最少要移動矩形的次數。

範例輸入	範例輸出
1 3 6 2 4 9	5 7

試題來源：ICPC 2019-2020 North-Western Russia Regional Contest

線上測試：計蒜客 A2270，Gym 102411A

❖ 試題解析

由於初始時，長矩形 $1 \times b$ 和短矩形 $1 \times a$ 是靠左對齊，所以開始要移動短矩形、且短矩形能移動的最大距離是 $b-a$。此後，每次長矩形和短矩形能移動的最大距離也是 $b-a$，對於 $2 \times n$ 大小的矩形盒子，長矩形要移動的距離是 $n-b$，所以，長矩形和短矩形交替移動，長矩形的最少移動次數是 $\left\lceil \frac{n-b}{b-a} \right\rceil$，而短矩形的最少移動次數是 $\left\lceil \frac{n-b}{b-a} \right\rceil + 1$。所以，Amelia 最少要移動矩形的次數是 $2 \times \left\lceil \frac{n-b}{b-a} \right\rceil + 1$。

因為 a、b 和 n 是整數，而整數的除法運算是向下取整數，所以，在程式中，要判斷 $(n-b)\%(b-a)$ 是否為 0。如果不為 0，則 $(n-b)/(b-a)$ 要向上取整數，即 $(n-b)/(b-a)+1$。

碁峰 www.gotop.com.tw

❖ 參考程式

```
01  #include<iostream>
02  using namespace std;
03  int main()
04  {
05      int a,b,n;
06      cin>>a>>b>>n;
07      int ans=1;            // 開始要移動短矩形 1 次
08      n-=b;                 // 長矩形要移動的距離
09      int k=b-a;            // 每次能移動的最大距離
10      ans+=(n/k)*2;         // n/k 向下取整數
11      if(n%k)               // n/k 需要向上取整數
12      {
13          ans+=2;
14      }
15      cout<<ans<<endl;      // 最少要移動矩形的次數
16      return 0;
17  }
```

「2.1.2 Sum」是一個 if else 選擇結構的實作。

2.1.2 ▸ Sum

請你求出 1 ～ *n* 之間所有整數的總和。

輸入

輸入是一個絕對值不大於 10000 的整數 *n*。

輸出

輸出一個整數，該整數是所有在 1 ～ *n* 之間的整數的總和。

範例輸入	範例輸出
−3	−5

試題來源：ACM 2000 Northeastern European Regional Programming Contest (test tour)

線上測試：Ural 1068

碁峰 www.gotop.com.tw

❖ 試題解析

本題要求算出 1 ～ n 之間所有整數的總和，而 n 是一個絕對值不大於 10000 的整數。等差數列的求和公式為 $S_n=n\times a_1+\frac{n\times(n-1)}{2}\times d$，其中 a_1 為首項，d 為公差，$n\in\mathbf{N}$。如果 n 是大於 0 的正整數，則 $S_n=1+2+\cdots+n=\frac{n\times(n+1)}{2}$，否則 $S_n=\frac{n\times(n+1)}{2}+1$。

❖ 參考程式

```
01  #include <iostream>
02  using namespace std;
03  int main()
04  {
05      int n;              // 輸入值：絕對值不大於 10000 的整數 n
06      cin>>n;
07      int s=0;            // s：1 ~ n 之間的整數的總和
08      if(n>0)             // if else 選擇結構完成等差數列的求和公式
09          s=n*(1+n)/2;
10      else
11          s=n*(1-n)/2+1;
12      cout<<s<<endl;      // 輸出所有在 1 ~ n 之間的整數的總和
13      return 0;
14  }
```

2.2　迴圈結構

迴圈述句有 while 述句、do while 述句和 for 述句。

while 述句的一般形式為：

```
while ( 運算式 )
{
    迴圈主體
}
```


碁峰 www.gotop.com.tw

功能：先判斷運算式值的真假，若為真（非零），就執行迴圈主體，否則結束迴圈結構。允許 while 述句的迴圈主體中包含另一個 while 述句，形成迴圈的巢狀。

do while 述句用來建構 UNTIL 迴圈結構，也多用於迴圈次數事先不確定的問題。其一般形式為：

```
do{
    迴圈主體
} while ( 運算式 );
```

功能：先執行一次迴圈主體，再判斷運算式的真假。若運算式為真，則繼續執行迴圈主體，一直到運算式為假時結束迴圈結構。注意 while 後面的「;」號不能少。由此可以看出，對於同一個問題，可以用 WHEN 迴圈，也可以用 UNTIL 迴圈，do while 的迴圈主體至少要被執行一次。

for 述句的一般形式為：

```
for( 運算式 1; 運算式 2; 運算式 3)
    迴圈主體
```

for 述句的執行過程為：

1. 第 1 步：求解運算式 1。

2. 第 2 步：求解運算式 2，若其值為真，則執行迴圈主體，求解運算式 3，繼續第 2 步；若運算式 2 值為假，則結束迴圈。

「2.2.1 Back to High School Physics」提供 while 述句的實作。

2.2.1 ► Back to High School Physics

一個粒子有初速度和加速度。如果在 t 秒時這個粒子的速度為 v，在 $2t$ 秒時這個粒子的總位移是多少？

輸入

輸入的每行提供兩個整數。每行構成一個測試案例，這兩個整數表示 v（$-100 \le v \le 100$）和 t（$0 \le t \le 200$）的值（t 表示粒子在 t 秒時的速度為 v）。

碁峰 www.gotop.com.tw

輸出

對於輸入的每行，在一行中輸出一個整數，為在 $2t$ 秒時這個粒子的總位移。

範例輸入	範例輸出
0 0 5 12	0 120

試題來源：BUET/UVA Oriental (WF Warmup) Contest 1

線上測試：UVA 10071

❖ 試題解析

粒子的加速度恆定，輸入 v 和 t，其中 v 是在時間點 t 的粒子的速度，求在時間點 $2t$ 粒子的位移。

本題涉及高中物理知識，分析如下。假設粒子的初速度為 v_0，加速度為 a，則在時間點 t，粒子的速度 $v=v_0+at$。根據位移公式 $s=v_0t+\frac{1}{2}at^2$，其中 s 是位移，v_0 是初速度，a 是加速度，在時間點 $2t$ 粒子的位移 $s=2v_0t+\frac{1}{2}a(2t)^2=2v_0t+2at^2=2t(v_0+at)=2vt$。

所以，本題迴圈輸入 v 和 t，計算 $2vt$，並輸出。

❖ 參考程式

```
01  #include <stdio.h>
02  int main(void)
03  {
04      int v, t;
05      while(scanf("%d%d", &v, &t) !=EOF)
06          printf("%d\n",2 * v * t);
07      return 0;
08  }
```

「2.2.2 Can You Solve It ?」提供 for 述句的實作。

碁峰 www.gotop.com.tw

2.2.2 ► Can You Solve It ?

請參見圖 2.2-1。在這張圖中，每個圓點都有一個笛卡兒座標系的座標。可以沿著由箭頭所表示的路徑從一個圓點到另一個圓點。從一個源點到一個目標點，所需要走的總步數 = 路徑通過的中間點的數目 +1。

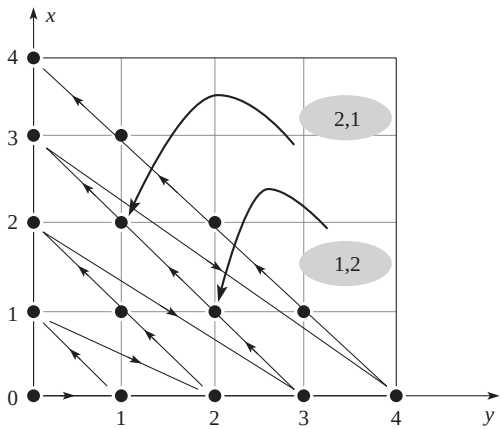

圖 2.2-1

如圖 2.2-1 所示，要從 (0, 3) 到 (3, 0)，就必須通過兩個中間點 (1, 2) 和 (2, 1)。所以，在這種情況下，所需要走的總步數是 2+1=3。本題要求計算從一個已知的源點到一個已知的目標點所需的步數。本題設定，對於所有的箭頭，不能走相反的方向。

輸入

輸入的第一行提供了要處理的測試案例數 n（$0<n\leq 500$）。接下來的 n 行每行提供 4 個整數（0≤ 每個整數 ≤100000），第一對整數表示源點的座標，另一對整數表示目標點的座標。座標以 (x, y) 形式列出。

輸出

對於每個測試案例，程式先輸出測試案例編號，然後輸出從源點到目標點所需的步數。本題設定可以從源點到達目標點。

碁峰 www.gotop.com.tw

範例輸入	範例輸出
3 0 0 0 1 0 0 1 0 0 0 0 2	Case 1: 1 Case 2: 2 Case 3: 3

試題來源：The FOUNDATION Programming Contest 2004

線上測試：UVA 10642

❖ 試題解析

本題提供了二維平面上整數點的座標，並用如圖 2.2-1 所示的箭頭的路徑將這些整數點連接起來；提供兩個二維平面座標點，計算從源點到目標點所需的步數。

對二維平面上的任一整數點，按箭頭所標示的順序，可以計算出 (0, 0) 到該點所需的步數。根據題意，同一層的整數點為右下至左上的斜線上的整數點，前 4 層的整數點的座標按箭頭所標示的順序如下：

(0, 0) →
(0, 1) → (1, 0) →
(0, 2) → (1, 1) → (2, 0) →
(0, 3) → (1, 2) → (2, 1) → (3, 0)

例如，要計算 (0, 0) 到 (2, 1) 所需的步數，(2, 1) 在第 4 層，該層的每一個整數點的 x 座標和 y 座標之和都是 3。從 (0, 0) 走完前 3 層，到第 4 層的第一個座標點 (0, 3) 所需的步數是 1+2+3=6。然後，從 (0, 3) 到 (2, 1) 需要走 2 步，恰好為 (2, 1) 的 x 座標值。所以 (0, 0) 到 (2, 1) 所需的步數為 6+2=8。

由上述實例，可以推出計算從 (0, 0) 到 (x, y) 所需步數的公式為：

$[1+2+3+\cdots+(x+y)]+x=(x+y+1)\times(x+y)/2+x$。

碁峰 www.gotop.com.tw

所以，對每個測試案例，先計算 (0, 0) 到源點所需的步數，以及 (0, 0) 到目標點所需的步數。然後，後者減去前者，即為源點到目標點所需的步數。

根據測試案例數，用 for 迴圈處理每個測試案例。

❖ 參考程式

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n;                                    // n：測試案例數目
    scanf("%d", &n);
    for(int k=1; k <=n; k++) {
        int x1, y1, x2, y2;                   // 源點和目標點的座標
        scanf("%d%d%d%d", &y1, &x1, &y2, &x2);
        int t1=(x1 + y1) * (x1 + y1 + 1) / 2 + y1;
        int t2=(x2 + y2) * (x2 + y2 + 1) / 2 + y2;
        printf("Case %d: %d\n", k, t2 - t1);  // 輸出源點到目標點所需的步數
    }
    return 0;
}
```

迴圈述句可以巢狀，「2.2.3 Gold Coins」和「2.2.4 The Hotel with Infinite Rooms」提供了迴圈巢狀的實作。

2.2.3 ▶ Gold Coins

國王要給他的忠誠騎士支付金幣。在他服務的第一天，騎士將獲得一枚金幣。在接下來的兩天的每一天（服務的第二和第三天），騎士將獲得 2 枚金幣。在接下來的 3 天的每一天（服務的第四、第五和第六天），騎士將獲得 3 枚金幣。在接下來的 4 天的每一天（服務的第七、第八、第九和第十天），騎士將獲得 4 枚金幣。這種支付模式將無限期地繼續下去：在連續 N 天的每一天獲得 N 枚金幣之後，在下一個連續的 N+1 天的每一天，騎士將獲得 N+1 枚金幣，其中 N 是任意的正整數。

請編寫程式，在已知天數的情況下，求出國王要支付給騎士的金幣的總數（從第一天開始計算）。

碁峰 www.gotop.com.tw

輸入

輸入至少一行，至多 21 行。每行提供問題的一個測試資料，即一個整數（範圍為 1 ～ 10000）表示天數。一行提供 0 表示輸入結束。

輸出

對於輸入中提供的每個測試案例，輸出一行。每行先提供在輸入中提供的天數，後面是一個空格，然後是在這些天數中，從第一天開始計算總共要支付給騎士的金幣數。

範例輸入	範例輸出
10	10 30
6	6 14
7	7 18
11	11 35
15	15 55
16	16 61
100	100 945
10000	10000 942820
1000	1000 29820
21	21 91
22	22 98
0	

試題來源：ACM Rocky Mountain 2004

線上測試：POJ 2000，ZOJ 2345，UVA 3045

❖ 試題解析

設 n 為總天數，這 n 天可以分成若干連續的時間段，第 i 個時間段為 i 天，每天獎勵 i 個金幣，則在這 i 天內共獎勵 $i \times i$ 個金幣。假設 ans 為獎勵的金幣總數，m 為當前天數。

本題的參考程式為雙重迴圈。

碁峰 www.gotop.com.tw